Models, Patterns and Generators

Panel on
Future of Software and Software Research

Gabor Karsai
ISIS
Vanderbilt University

Trends in Software Engineering

- **Move towards *higher-level, model-oriented* approaches**
  - **OO, UML, OMG's MDA,…**
- **Design patterns to codify working solutions**
  - **GOF, PLoP, …**
- **Domain-specific languages and generative techniques for product lines**
  - **Amortize the cost of developing domain-specific tools over the products developed**
- **Systems built as networks of systems**
  - **Software as web services**
- **Software in a physical context**
  - **Embedded systems**

Towards a ***Model-based*** Approach

Traditional software approach:
Design -> Implementation

Model-based approach:
Design Modeling => Generated Implementation

Metamodeling

Domain-specific, multi-aspect, yet integrated models of the problem, its context, and the solution

Whenever possible, the implementation (or parts of it) should be generated from the models.

Modeling languages for specific domains must be precisely defined using metamodeling languages, as well as the translation of their abstractions.

"Good practices" are codified in the form of design patterns: templates that can be reused in many contexts

The needs

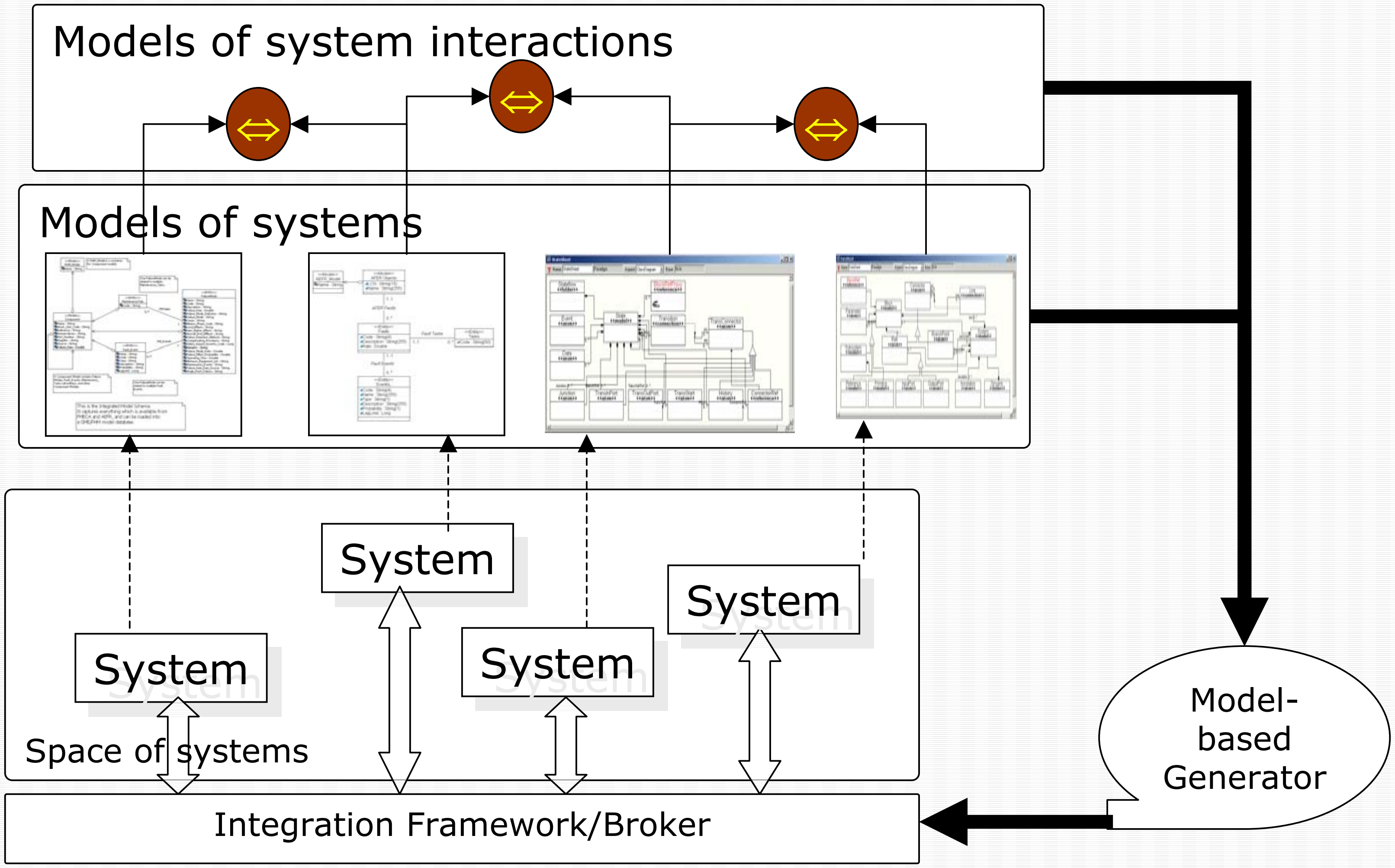
- Easy-to-construct/customize…
  1. Domain specific modeling languages and modeling environments
  2. Model translation/transformation tools
  3. Development environments with support for large-scale development
  4. Generic analysis tools
- Tools to describe modeling languages and their translation/interpretation:
  - Meta-modeling technology

Specific application of the model-based approach

Challenge: Integration technology

- Software is needed for constructing systems of systems
- Approach:
  - Model component systems, interactions, and constraints
  - Generate “glue”

Integration via models



Models of system interactions
⇔
⇔
⇔
Models of systems
System
System
System
System
Space of systems
Integration Framework/Broker
Model-based Generator